

Complexity, Decidability and Undecidability Results for Domain- Independent Planning

Kutluhan Erol, Dana S. Nau,
and V.S. Subrahmanian

Presented by: Matthias Gauger
CS 7612: AI Planning

Introduction

- Planning is intractable in general
- Idea: Try to identify constraints that lead to efficient planning.
- Question: How does complexity depend on the nature of the planning operators?
- Analyze STRIPS-style operators
- Transfer of decidability results from logic programming to planning

Basic definitions

- First-order language
- A planning operator is a 4-tuple $(\text{Name}(\alpha), \text{Pre}(\alpha), \text{Add}(\alpha), \text{Del}(\alpha))$
- A planning domain is a pair $P = (S_0, O)$
- A planning problem instance is a triple $P = (S_0, O, G)$
- A conditional operator α describes an action whose effects depend on the input situation.

Two decision problems

- PLAN EXISTENCE: “Given a planning problem instance $P = (S_0, O, G)$, is there a plan in P that achieves G ?”
- PLAN LENGTH: “Given a planning problem instance $P = (S_0, O, G)$ and an integer k encoded in binary, is there a plan in P of length k or less that achieves G ?”
 - * In reality, you normally want to find the shortest plan...

A short reminder...

- A language is called *decidable* if there exists a method - any method at all - to determine whether a given word belongs to that language or not.
- A language L is semidecidable if there exists an algorithm for membership that will always correctly answer “yes”, but may diverge before answering “no”.

Decidability of domain-independent planning

Allow function symbols?	Allow infinitely many constant symbols?	Infinite initial states?	Allow delete lists and / or negated preconditions?	PLAN EXISTENCE	
yes	yes / no	yes / no	yes / no / no	semidecidable	
	no	no	no*	decidable	
no	yes	yes	yes / no	semidecidable	
		no	yes	semidecidable	
	no	no	no	no	decidable
		no	no	yes / no	decidable

Interesting insights (1)

- Plan existence is, in general, undecidable if the language is allowed to contain function symbols.
 - ★ A language that is allowed to contain function symbols can contain infinitely many ground terms.
- If the language doesn't contain function symbols, then plan existence is, in general, decidable.
 - ★ only finitely many ground terms

Interesting insights (2)

- The results are independent of whether or not the operators are fixed in advance and whether or not the operators are allowed to have conditional effects.
- Many results obtained using analogy to logic programming
 - * Planning without delete lists is practically equivalent to logic programming.

A short reminder...

	<i>Class of problems solvable in ...</i>
<i>EXPSPACE</i>	exponential space
<i>NEXPTIME</i>	non-deterministic exponential time
<i>EXPTIME</i>	deterministic exponential time
<i>PSPACE</i>	polynomial space
<i>NP</i>	non-deterministic polynomial time
<i>P</i>	deterministic polynomial time
<i>NLOGSPACE</i>	non-deterministic logarithmic space

Complexity of domain-independent planning

Language restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Plan existence	Plan length
datalog (no function symbols, only finitely many constant symbols)	given in the input	yes	yes / no	EXPSpace-comp.	NEXPTIME-comp.
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.
			no	EXPTIME-comp.	NEXPTIME-comp.
			no*	PSPACE-comp.	PSPACE-comp.
	fixed in advance	yes	yes / no	PSPACE	PSPACE
		no	yes	NP	NP
			no	P	NP
			no / no	NLOGSPACE	NP
propositional (all predicates are 0-ary)	given in the input	yes	yes / no	PSPACE-comp.	PSPACE-comp.
		no	yes	NP-comp.	NP-comp.
			no	P	NP-comp.
			no / no	NLOGSPACE-comp.	NP-comp.
	fixed in advance	yes / no	yes / no	constant time	constant time

Complexity of domain-independent planning

Language restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Plan existence	Plan length
datalog (no function symbols, only finitely many constant symbols)	given in the input	yes	yes / no	EXPSPACE-comp.	NEXPTIME-comp.
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.
			no	EXPTIME-comp.	NEXPTIME-comp.
			no*	PSPACE-comp.	PSPACE-comp.
	fixed in advance	yes	yes / no	PSPACE	PSPACE
		no	yes	NP	NP
			no	P	NP
			no / no	NLOGSPACE	NP
propositional (all predicates are 0-ary)	given in the input	yes	yes / no	PSPACE-comp.	PSPACE-comp.
		no	yes	NP-comp.	NP-comp.
			no	P	NP-comp.
			no / no	NLOGSPACE-comp.	NP-comp.
	fixed in advance	yes / no	yes / no	constant time	constant time

Complexity of domain-independent planning

Language restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Plan existence	Plan length
datalog (no function symbols, only finitely many constant symbols)	given in the input	yes	yes / no	EXPSPACE-comp.	NEXPTIME-comp.
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.
			no	EXPTIME-comp.	NEXPTIME-comp.
			no*	PSPACE-comp.	PSPACE-comp.
	fixed in advance	yes	yes / no	PSPACE	PSPACE
		no	yes	NP	NP
			no	P	NP
			no / no	NLOGSPACE	NP
propositional (all predicates are 0-ary)	given in the input	yes	yes / no	PSPACE-comp.	PSPACE-comp.
		no	yes	NP-comp.	NP-comp.
			no	P	NP-comp.
			no / no	NLOGSPACE-comp.	NP-comp.
	fixed in advance	yes / no	yes / no	constant time	constant time

Complexity of domain-independent planning

Language restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Plan existence	Plan length
datalog (no function symbols, only finitely many constant symbols)	given in the input	yes	yes / no	EXPSpace-comp.	NEXPTIME-comp.
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.
			no	EXPTIME-comp.	NEXPTIME-comp.
			no*	PSPACE-comp.	PSPACE-comp.
	fixed in advance	yes	yes / no	PSPACE	PSPACE
		no	yes	NP	NP
			no	P	NP
			no / no	NLOGSPACE	NP
propositional (all predicates are 0-ary)	given in the input	yes	yes / no	PSPACE-comp.	PSPACE-comp.
		no	yes	NP-comp.	NP-comp.
			no	P	NP-comp.
			no / no	NLOGSPACE-comp.	NP-comp.
	fixed in advance	yes / no	yes / no	constant time	constant time

Complexity of domain-independent planning

Language restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Plan existence	Plan length
datalog (no function symbols, only finitely many constant symbols)	given in the input	yes	yes / no	EXPSpace-comp.	NEXPTIME-comp.
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.
			no	EXPTIME-comp.	NEXPTIME-comp.
			no*	PSPACE-comp.	PSPACE-comp.
	fixed in advance	yes	yes / no	PSPACE	PSPACE
		no	yes	NP	NP
			no	P	NP
			no / no	NLOGSPACE	NP
propositional (all predicates are 0-ary)	given in the input	yes	yes / no	PSPACE-comp.	PSPACE-comp.
		no	yes	NP-comp.	NP-comp.
			no	P	NP-comp.
			no / no	NLOGSPACE-comp.	NP-comp.
	fixed in advance	yes / no	yes / no	constant time	constant time

Complexity of domain-independent planning

Language restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Plan existence	Plan length
datalog (no function symbols, only finitely many constant symbols)	given in the input	yes	yes / no	EXPSpace-comp.	NEXPTIME-comp.
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.
			no	EXPTIME-comp.	NEXPTIME-comp.
			no*	PSPACE-comp.	PSPACE-comp.
	fixed in advance	yes	yes / no	PSPACE	PSPACE
		no	yes	NP	NP
			no	P	NP
			no / no	NLOGSPACE	NP
propositional (all predicates are 0-ary)	given in the input	yes	yes / no	PSPACE-comp.	PSPACE-comp.
		no	yes	NP-comp.	NP-comp.
			no	P	NP-comp.
			no / no	NLOGSPACE-comp.	NP-comp.
	fixed in advance	yes / no	yes / no	constant time	constant time

Complexity of domain-independent planning

Language restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Plan existence	Plan length
datalog (no function symbols, only finitely many constant symbols)	given in the input	yes	yes / no	EXPSpace-comp.	NEXPTIME-comp.
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.
			no	EXPTIME-comp.	NEXPTIME-comp.
			no*	PSPACE-comp.	PSPACE-comp.
	fixed in advance	yes	yes / no	PSPACE	PSPACE
		no	yes	NP	NP
			no	P	NP
			no / no	NLOGSPACE	NP
propositional (all predicates are 0-ary)	given in the input	yes	yes / no	PSPACE-comp.	PSPACE-comp.
		no	yes	NP-comp.	NP-comp.
			no	P	NP-comp.
			no / no	NLOGSPACE-comp.	NP-comp.
	fixed in advance	yes / no	yes / no	constant time	constant time

Complexity of domain-independent planning

Language restrictions	How the operators are given	Allow delete lists?	Allow negated preconditions?	Plan existence	Plan length
datalog (no function symbols, only finitely many constant symbols)	given in the input	yes	yes / no	EXPSpace-comp.	NEXPTIME-comp.
		no	yes	NEXPTIME-comp.	NEXPTIME-comp.
			no	EXPTIME-comp.	NEXPTIME-comp.
			no*	PSPACE-comp.	PSPACE-comp.
	fixed in advance	yes	yes / no	PSPACE	PSPACE
		no	yes	NP	NP
			no	P	NP
			no / no	NLOGSPACE	NP
propositional (all predicates are 0-ary)	given in the input	yes	yes / no	PSPACE-comp.	PSPACE-comp.
		no	yes	NP-comp.	NP-comp.
			no	P	NP-comp.
			no / no	NLOGSPACE-comp.	NP-comp.
	fixed in advance	yes / no	yes / no	constant time	constant time

Interesting insights

- Computational complexity varies from constant time to EXPSpace-complete.
- Operators with conditional effects do not affect the complexity results.
- In most cases, the complexity in the datalog case is exactly one level higher than the complexity in the corresponding propositional case.
- Delete lists are more powerful than negated preconditions.

Reference

- K. Erol, D. Nau, V. Subrahmanian. “Complexity, Decidability and Undecidability Results for Domain-Independent Planning”. *Artificial Intelligence*, 76(1-2), pp. 75-88, 1995.